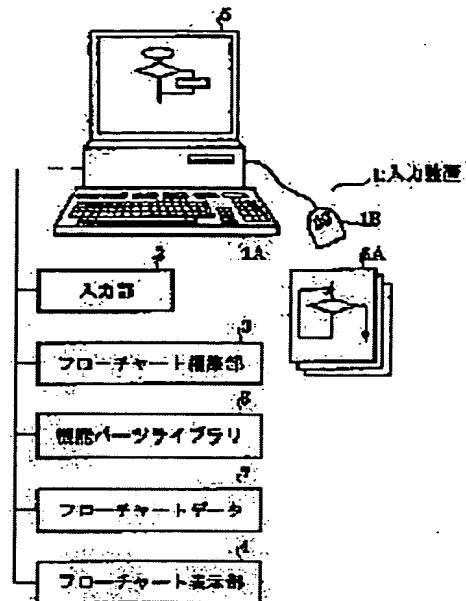


(11)Publication number : 11-296357
(43)Date of publication of application : 29.10.1999

(21)Application number : 10-112756 (71)Applicant : OKI ELECTRIC IND CO LTD
(22)Date of filing : 08.04.1998 (72)Inventor : YAMADA ZENSHI

(57)Abstract:

SOLUTION: A user generates a flow chart while viewing a display device 5 through the use of the input device 1. The device 1 is constituted of the parts such as a keyboard 1A and a mouse 1B, for displaying the intention of the user in the device. In this case, the device 1 selects the optional function parts stored in a function parts library 6 and designates that they are displayed at the prescribed positions on a display device 5. Then, an input part 2 analyzes the operation of the user, judges the propriety of the operation and gives a proper indication to a flow chart editing part 3. The part 3 executes a processing for editing flow chart data 7 in accordance with the intention of the user.



<http://www19.ipdl.jpo.go.jp/PA1/result/detail/.../wAAACna4dJDA411296357P1.ht> 2004/04/27

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-296357

(43) 公開日 平成11年(1999)10月29日

(51) Int.Cl.⁹

G 0 6 F 9/06

識別記号

5 3 0

F I

G 0 6 F 9/06

5 3 0 H

5 3 0 W

審査請求 未請求 請求項の数13 F D (全 18 頁)

(21) 出願番号

特願平10-112756

(22) 出願日

平成10年(1998)4月8日

(71) 出願人 000000295

沖電気工業株式会社

東京都港区虎ノ門1丁目7番12号

(72) 発明者 山田 善嗣

東京都港区虎ノ門1丁目7番12号 沖電気
工業株式会社内

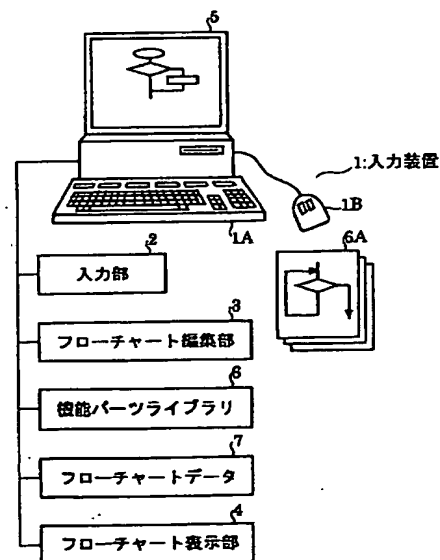
(74) 代理人 弁理士 佐藤 幸男 (外1名)

(54) 【発明の名称】 フローチャート入力装置

(57) 【要約】

【解決手段】 表示装置5にフローチャートを描く際、機能パーツライブラリ6を使用する。機能パーツは、フローチャート中に使用されて一定の処理を完結する手順を表現する図形パーツ群により構成される。機能パーツ群を、許された場所に許された順で挿入し、フローチャートを組み立てる。

【効果】 論理的に矛盾のないイベントドリブンのフローチャートを容易に作成でき、メカトロニクス製品のソフトウェア開発が容易になる。



本発明のフローチャート入力装置ブロック図

【特許請求の範囲】

【請求項 1】 フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、機能パーツのイメージを複数格納した機能パーツライブラリと、

前記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定する入力装置と、

入力装置により選択された前記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するフローチャート編集部とを備えたことを特徴とするフローチャート入力装置。

【請求項 2】 請求項 1 に記載のフローチャート入力装置において、

各機能パーツには、予め他の機能パーツを挿入して配置することができる位置を表示する空パーツが含まれることを特徴とするフローチャート入力装置。

【請求項 3】 請求項 2 に記載のフローチャート入力装置において、

入力装置により指定された機能パーツの挿入位置が、既に表示装置に表示された機能パーツの空パーツ上でないときは、挿入を拒絶する挿入判定部を備えたことを特徴とするフローチャート入力装置。

【請求項 4】 請求項 3 に記載のフローチャート入力装置において、各機能パーツの空パーツ部分に挿入することができる機能パーツを予め限定する挿入判定マトリクスを備え、挿入判定部は、前記挿入判定マトリクスを参照して、入力装置により選択された機能パーツの挿入を拒絶するかどうか判定することを特徴とするフローチャート入力装置。

【請求項 5】 請求項 1 に記載のフローチャート入力装置において、

フローチャート編集部は、

既に表示装置に表示されたフローチャート中に、新たな機能パーツが挿入されたときは、次に他の機能パーツを挿入して配置することができる位置を表示する空パーツを、前記フローチャート中の機能パーツ間に配置することを特徴とするフローチャート入力装置。

【請求項 6】 請求項 1 に記載のフローチャート入力装置において、

各機能パーツには、予め、プログラムの流れを示す矢印を接続することができる位置を表示する接続点マークが含まれることを特徴とするフローチャート入力装置。

【請求項 7】 請求項 6 に記載のフローチャート入力装置において、

表示装置上に表示された任意の機能パーツに含まれる矢印の接続位置が、入力装置により指定されたとき、既に表示装置に表示された機能パーツの接続点マーク上でな

い場合にその接続を拒絶する接続判定部を備えたことを特徴とするフローチャート入力装置。

【請求項 8】 請求項 7 に記載のフローチャート入力装置において、

接続判定部は、指定された矢印の接続位置が既に表示装置に表示されている機能パーツとの関係から所定の基準を満たさないときは、入力装置により指定された矢印の接続を拒絶することを特徴とするフローチャート入力装置。

10 【請求項 9】 請求項 1 に記載のフローチャート入力装置において、

フローチャート編集部は、

フローチャートデータを階層構造により表現し、状態を示すステップに関する情報と、状態遷移を示す矢印に関する情報とを含むデータ階層の下に、前記状態を示すステップの属性情報を含むデータ階層を設けるようにしたことを特徴とするフローチャート入力装置。

【請求項 10】 請求項 9 に記載のフローチャート入力装置において、

20 フローチャートデータ中の、状態を示すステップに関する情報と、状態遷移を示す矢印に関する情報とを含むデータ階層を選択するとともに、前記状態遷移の原因となるイベントをさらに下層のデータ階層から抽出して状態遷移図を生成する状態遷移図生成部を備えたことを特徴とするフローチャート入力装置。

【請求項 11】 請求項 1 に記載のフローチャート入力装置において、

各機能パーツに対応させて用意されたプログラムソースコードと、フローチャートデータとを編集して、フローチャートを実行するためのソースプログラムを自動生成するプログラム生成部を備えたことを特徴とするフローチャート入力装置。

【請求項 12】 フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、

予め、機能パーツのイメージを複数格納した機能パーツライブラリを用意して、

入力装置により、前記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定したとき、

40 選択された前記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するように動作する、フローチャート入力用のコンピュータプログラムを記録した記録媒体。

【請求項 13】 フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、

予め、機能パーツのイメージを複数格納した機能パーツライブラリを用意して、

50 入力装置により、前記機能パーツライブラリに格納され

た任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定したとき、
 選択された前記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するとともに、
 各機能パーツに対応させて用意されたプログラムソースコードと、フローチャートデータとを編集して、フローチャートを実行するためのソースプログラムを自動生成するように動作する、フローチャート入力用のコンピュータプログラムを記録した記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ソフトウェア開発、特にメカトロニクス製品の開発等を支援するためのドキュメントからプログラムへの自動変換可能なCASE (computer aided software engineering) ツールとなり得る、フローチャート入力装置に関する。

【0002】

【従来の技術】メカトロニクス製品は、論理回路、機構、ソフトウェアの3要素からなる。そのため、機構設計者、ソフトウェア設計者、回路設計者の三者が共同して製品開発を進めなければならない。通常、機構をどのように制御し、どのように動作させるかというイメージは、その機構を設計した設計者が持っている。しかしながら、一般に機構設計者はそのイメージをソフトウェアとして実現する技術を持たない。一方、ソフトウェア設計者は、ソフトウェアを実現する技術を持っても、新たに設計された機構の制御方法や動作は機構設計者から聞かなければならない。従って、機構設計者は、機構の制御、動作イメージをソフトウェア設計者に伝達するためのドキュメントを作成する必要がある。

【0003】ソフトウェア設計者は機構設計者から受け取ったドキュメントをもとに、ソフトウェアを開発する。このため、そのドキュメントには矛盾や曖昧さがなく、プログラマ的に見て正しいことが要求される。しかし、機構設計者はソフトウェア設計の技術を持たないため、こうしたドキュメントを作成するのは容易でない。従って、ソフトウェア設計の技術を持たない者でも正しいドキュメントを容易に描き得るツールが必要となる。

【0004】また、メカトロニクス製品のソフトウェアは、高度な処理速度性能が要求される。そのため、割り込みを多用したイベントドリブン指向のプログラムとなる。従って、機構作成者が作成するドキュメントも、現実のプログラムに即したイベントドリブン指向で記述されていることが望ましい。

【0005】即ち、メカトロニクス製品の開発に適用するドキュメント作成ツールには次のような要件が望まれる。1. ソフトウェアの知識がなくても、正しいドキュメントが作成できること。2. イベントドリブン指向の

記述が可能なこと。

【0006】一方、従来から行われているドキュメントの作成方法には以下のようなものがある。

1. 手作業でフローチャートを描く。
2. ドロー系ツールを使用してフローチャートを描く。
3. HCPチャート、PAD、HIPO、NSチャートと呼ばれるソフトウェア開発専用のCASEツールを使用する。
4. フローチャートを採用したCASEツールを使用する。

【0007】

【発明が解決しようとする課題】ところで、上記のような従来の技術には次のような解決すべき課題があった。フローチャートを手作業で描くには多大な工数が必要になる。また、記述の自由度が大きすぎて、フローチャートの正当性のチェックも人に頼らざるを得ない。更に、ソフトウェアの知識がないと、システムのサポートがなければ、正しいフローチャートを作成することが困難である。フローチャート用のテンプレートを用意したドロー系のツールを使用すれば、図形描画が容易になり、入力工数は削減される。しかし、その他の問題点は手作業と変わらない。

【0008】一方、HCPチャート、PAD、HIPO、NSチャート等のCASEツールは、フローチャートを用いないため、上記の問題が解決される。また、この手法自体、フローチャートに比べて適切に自由度が制限され、比較的正しいドキュメントを書き易い。また、プログラムへの自動変換も考慮して開発がされている。しかしながら、これらの手法は、まだ一般的でなく、標準化もなされておらず、特にソフトウェア設計者でない者にとって馴染みが薄いという問題がある。

【0009】フローチャートを使用したCASEツールは多くは存在しない。特に、プログラムに自動変換可能なものはまれである。また、フローチャートという手法自体の問題点から、g o t o レスの構造化されたプログラムに変換できるものは見当たらない。即ち、従来のCASEツールはソフトウェア設計者を対象にしたものが多く、プログラミングの知識がないと正しいドキュメントを作成することは困難である。また、これらのツールには組込み機器用即ちメカトロニクス製品用に特化されたものは少ない。そのため、イベントドリブン指向のドキュメントを作成できるものが少ない。

【0010】

【課題を解決するための手段】本発明は以上の点を解決するため次の構成を採用する。

〈構成1〉フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、機能パーツのイメージを複数格納した機能パーツライブラリと、上記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の

10

20

30

40

50

所定の位置に表示するように指定する入力装置と、入力装置により選択された上記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するフローチャート編集部とを備えたことを特徴とするフローチャート入力装置。

【0011】〈構成2〉構成1に記載のフローチャート入力装置において、各機能パーツには、予め他の機能パーツを挿入して配置することができる位置を表示する空パーツが含まれることを特徴とするフローチャート入力装置。

【0012】〈構成3〉構成2に記載のフローチャート入力装置において、入力装置により指定された機能パーツの挿入位置が、既に表示装置に表示された機能パーツの空パーツ上でないときは、挿入を拒絶する挿入判定部を備えたことを特徴とするフローチャート入力装置。

【0013】〈構成4〉構成3に記載のフローチャート入力装置において、各機能パーツの空パーツ部分に挿入することができる機能パーツを予め限定する挿入判定マトリクスを備え、挿入判定部は、上記挿入判定マトリクスを参照して、入力装置により選択された機能パーツの挿入を拒絶するかどうか判定することを特徴とするフローチャート入力装置。

【0014】〈構成5〉構成1に記載のフローチャート入力装置において、フローチャート編集部は、既に表示装置に表示されたフローチャート中に、新たな機能パーツが挿入されたときは、次に他の機能パーツを挿入して配置することができる位置を表示する空パーツを、上記フローチャート中の機能パーツ間に配置することを特徴とするフローチャート入力装置。

【0015】〈構成6〉構成1に記載のフローチャート入力装置において、各機能パーツには、予め、プログラムの流れを示す矢印を接続することができる位置を表示する接続点マークが含まれることを特徴とするフローチャート入力装置。

【0016】〈構成7〉構成6に記載のフローチャート入力装置において、表示装置上に表示された任意の機能パーツに含まれる矢印の接続位置が、入力装置により指定されたとき、既に表示装置に表示された機能パーツの接続点マーク上でない場合にその接続を拒絶する接続判定部を備えたことを特徴とするフローチャート入力装置。

【0017】〈構成8〉構成7に記載のフローチャート入力装置において、接続判定部は、指定された矢印の接続位置が既に表示装置に表示されている機能パーツとの関係から所定の基準を満たさないときは、入力装置により指定された矢印の接続を拒絶することを特徴とするフローチャート入力装置。

【0018】〈構成9〉構成1に記載のフローチャート入力装置において、フローチャート編集部は、フローチャートデータを階層構造により表現し、状態を示すステ

ップに関する情報と、状態遷移を示す矢印に関する情報とを含むデータ階層の下に、上記状態を示すステップの属性情報を含むデータ階層を設けるようにしたことを特徴とするフローチャート入力装置。

【0019】〈構成10〉構成9に記載のフローチャート入力装置において、フローチャートデータ中の、状態を示すステップに関する情報と、状態遷移を示す矢印に関する情報とを含むデータ階層を選択するとともに、上記状態遷移の原因となるイベントをさらに下層のデータ階層から抽出して状態遷移図を生成する状態遷移図生成部を備えたことを特徴とするフローチャート入力装置。

【0020】〈構成11〉構成1に記載のフローチャート入力装置において、各機能パーツに対応させて用意されたプログラムソースコードと、フローチャートデータとを編集して、フローチャートを実行するためのソースプログラムを自動生成するプログラム生成部を備えたことを特徴とするフローチャート入力装置。

【0021】〈構成12〉フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、予め、機能パーツのイメージを複数格納した機能パーツライブラリを用意して、入力装置により、上記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定したとき、選択された上記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するように動作する、フローチャート入力用のコンピュータプログラムを記録した記録媒体。

【0022】〈構成13〉フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、予め、機能パーツのイメージを複数格納した機能パーツライブラリを用意して、入力装置により、上記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定したとき、選択された上記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するとともに、各機能パーツに対応させて用意されたプログラムソースコードと、フローチャートデータとを編集して、フローチャートを実行するためのソースプログラムを自動生成するように動作する、フローチャート入力用のコンピュータプログラムを記録した記録媒体。

【0023】

【発明の実施の形態】以下、本発明の実施の形態を具体例を用いて説明する。

〈具体例1〉図1は、本発明のフローチャート入力装置を示すブロック図である。図に示すようにこの装置は、入力装置1、入力部2、フローチャート編集部3、フローチャート表示部4、表示装置5、機能パーツライブラリ6、フローチャートデータ7等から構成される。

【0024】ユーザは、入力装置1を用いて表示装置5を見ながらフローチャートを作成する。入力装置1は、キーボード1Aやマウス1B等、ユーザの意志を装置に指示するための部品により構成される。入力部2は、ユーザの操作を解釈し、その操作の可否を判断し、フローチャート編集部3に適切な指示を出す処理を行う部分である。フローチャート編集部3は、ユーザの意志に従ってフローチャートデータ7を編集する処理を行う部分である。

【0025】フローチャート表示部4は、フローチャートデータ7の更新状態を監視して、編集されて更新されたフローチャートデータをそのつど表示装置5に表示する制御を行う部分である。更に、フローチャート表示部4は、ユーザに対し操作の可否を知らせる処理も行う。表示装置5は、ブラウン管ディスプレイや液晶表示盤等から成り、ユーザに編集結果を知らせるための装置である。機能パーツライブラリ6は、後で詳しく説明するように、ユーザがフローチャートを構築する上でその部品となるイメージや情報を格納する部分である。フローチャートデータ7は、フローチャート編集部3によって編集された、後で説明するような構造のデータから構成される。

【0026】以上の装置は、例えばワークステーションやパーソナルコンピュータに上記の機能ブロックを実現するコンピュータプログラムをインストールして実現される。入力部2、フローチャート編集部3、フローチャート表示部4等は、それぞれの機能を実現するプログラムモジュールにより構成される。また、機能パーツライブラリ6やフローチャートデータ7は、コンピュータ中に設けられた記憶装置の内部に蓄積された所定の構成のデータにより実現される。

【0027】図2に、図1に示した表示装置5に表示されたウインドウの概略図を示す。この画面は、フローチャート入力作業を行う場合に表示される画面で、メインウインドウ11とパーツパレット12とから構成される。メインウインドウ11には、フローチャートを作成し表示するためのフローチャート表示エリア13と、ユーザが各種の操作を指示するためのメインメニュー14と、マウスを用いてクリックし具体的な操作を直接指示するツールボタン群15を備えている。パーツパレット12には、図1に示した機能パーツライブラリ6の様々な機能パーツ6Aを選択するボタンが並んでいる。なお、ツールボタン群15や機能パーツ6Aを選択するボタンの細部の図示は、ここでは省略している。

【0028】従来は、パーツパレットにより菱形や四角形、矢印といった図形パーツのテンプレートを用意し、こうしたフローチャートを構築するための部品として使用していた。しかしながら、図形パーツ自体はプログラムとして意味を持たず、フローチャート作成上の自由度が大きすぎる。そこで、本発明では次のような機能パー

ツを単位としてフローチャートの作成を補助する。即ち、図2のパーツパレット12は、機能パーツを1単位の部品としている。

【0029】図3に、機能パーツの具体例説明図を示す。図3の(a)～(f)には、本発明でフローチャート作成に使用する機能パーツの様々な例を示した。機能パーツは、それぞれ、プログラム中に使用されて一定の処理を完結する手順を表現するための図形パーツ群から構成される。図形パーツというのは、既に説明したように、菱形や四角形、矢印等のことである。

【0030】例えば、図3(a)は、「ステップ」と呼ばれる機能パーツである。これは、待機ループ中でイベントが発生すると、別の状態に遷移する機能を持ち、待機ループ16中に判断用の菱形17を配置したものである。これは状態遷移図の1つの状態に相当する。また、状態遷移マトリクス表の列に相当する。このパーツが通常のループと異なるのは、通常のループが処理の繰り返しであるのに対し、ステップの機能パーツはタスクのウェイト状態に移行する点である。

【0031】イベント発生チェックはオペレーティングシステムに任される。オペレーティングシステムはイベントの発生を検知すると、そのイベントに対応した処理を実行する。イベントドリブン指向のプログラムとは、こうしたステップの組み合わせで記述されたプログラムを指している。なお、同一の状態で複数種類のイベント待ちを行うことがあるから、待機ループ16中に後から任意の数の菱形17を挿入することができる。

【0032】図3(b)に示す分岐という機能パーツは、菱形17の部分で一定の条件判断をし、2種類のルートに分岐する機能を持つ。各分岐後のルートには必要に応じて後から任意の処理が挿入される。これも入口から出口までの間に一定の完結した処理を実行する機能を持つ。(c)は、ケース文の機能パーツである。菱形17の部分で条件判断をし、ケースに応じて3つ以上のルートに分岐する。各ルートでは、それぞれケースに応じた処理が実行される。長方形18は、その処理を示す。

【0033】図3(d)は、端子の機能パーツである。プログラムの開始時や終了時に、このパーツ19が使用される。(e)は前判定ループ、(f)は後判定ループである。前判定ループは、菱形17で一定の判断を行い、例えば条件を満たさない場合には長方形18を含む処理ループを繰り返し、条件を満たした場合にはこの機能パーツの処理から抜ける。後判定ループの場合、予め長方形18の処理を実行し、菱形17で条件判断を行う。条件を満たさない場合には、長方形18を含む処理ループを繰り返す。条件を満たした場合にはこの機能パーツの処理を抜ける。

【0034】以上のように、いずれの機能パーツもプログラムの正しい意味を持ち、一定の処理を完結し、フローチャート構築の単位となる。ユーザは、こうした機

10

20

30

40

50

能パーツを配列していくことによってフローチャートを組み立てるので、プログラムの正しいフローチャートを記述することができる。また、これらの機能パーツによる処理の内容はそれぞれプログラムソースコードに対応させることができる。

【0035】例えば(a)は、イベントをそのままオペレーティングシステムに伝えるメッセージ群により記述できる。(b)はいわゆるif~Then, Else文により記述される。また、(c)はcase文により記述される。(d)の端子はend文である。(e)や

(f)は、if文やfor文により記述される。

【0036】図4には、具体的なソフトウェア設計者がフローチャートを設計する場合の設計例説明図を示す。例えば、図4(a)に示すような機構についてフローチャートを作成するものとする。図の(a)に示す機構は、搬送モータ21を用いてコンベアー22を駆動し、工作物23を図の右側から左側の方向に搬送するように動作する。ここで、工作物23は、コンベアー22上を搬送される途中、第1センサ24や第2センサ25によって位置検出される。第1センサ24が工作物23の通過を検出すると一定の処理が実行される。更に、第2センサ25が工作物23を検出すると、別の処理が実行される。このような場合、図の(b)に示すようなフローチャートを作成する。

【0037】即ち、図4(b)の処理S1において、モータがスタートすると、処理S2において、第1センサ24が工作物23の通過というイベント待ち状態に入る。第1センサが工作物23を検出すると、S3の処理1に進む。そして、その後処理S4に進み、第2センサが工作物23の通過待ちというイベント待ち状態に移る。第2センサが工作物23の通過を検出すると、S5に進んで処理2が実行される。

【0038】実際のメカトロニクス製品では、様々な複雑な制御が行われるが、例えば図4(b)のような制御を行う場合に、第1センサの監視や第2センサの監視には、図3(a)を用いて説明したステップの機能パーツをそのまま利用する。後は、図3には例示しなかったが、長方形のみの処理を示す機能パーツを組み合わせる。本発明の装置は、こうしてフローチャートを作成する場合に活用される。

【0039】図5に、フローチャート作成とプログラムへの変換例説明図を示す。この図を用いて、図1に示す装置の具体的なフローチャート入力操作を説明する。まず、ユーザが図2に示すような画面を図1に示す表示装置5に表示させる。そして、パーツパレット12から様々な機能パーツをドラッグしてフローチャート表示エリア13上にドロップする。この操作を繰り返すことによって、例えば図5(a)に示すようなフローチャートが描かれる。このフローチャートは、初期処理の機能パーツの次に図3(a)に示したステップの機能パーツを接

続し、予めステップの機能パーツに含まれているイベント1に更にイベント2の機能パーツを追加する。その後、イベント処理1の機能パーツとイベント処理2の機能パーツを接続する。

【0040】図1に示すマウス1Bの操作によって、このような機能パーツがドラッグされると、入力部2が機能パーツのドラッグ開始を検出する。ユーザは、フローチャート表示エリア13上の所定の位置を指定して、機能パーツをドロップする。入力部2は、機能パーツのドロップを検出する。そして、そのドロップ位置がフローチャート中のどの位置かを判断する。フローチャート中の適切な位置にその機能パーツが挿入された場合、これをフローチャート編集部3に通知する。

【0041】フローチャート編集部3は、機能パーツライブラリ6から該当する機能パーツのデータを取り出して、フローチャートデータ7の該当する部分に対応するデータを書き込む。フローチャート表示部4は、フローチャートデータ7が更新されたことを検出すると、表示装置5に対し該当する機能パーツのイメージを取り出して表示する。以上のようにして、フローチャートの入力編集が進められる。

【0042】図5(b)は、従来のドロー系のツールを使用する場合のフローチャート用図形パーツである。この図に示す菱形17は、これだけではプログラムの意味をなさない。図の(c)に示すように、条件判断の結果分岐するルートを示す矢印等が含まれて初めて、一定の処理を完結する手順が表現される。このような機能パーツ単位でフローチャートの作成を支援することによりフローチャートの作成が容易になり、かつ、正確になる。更に、図5(c)に示すように、この機能パーツはそのままif文に置き換えることができ、ソースプログラムの自動作成が可能になる。

【0043】〈具体例1の効果〉以上のように、プログラム作成を機能パーツ単位で行い、機能パーツのイメージを格納した機能パーツライブラリと、そのフローチャート中の位置を指定する入力装置と、フローチャートデータを編集するフローチャート編集部等によってプログラムの誤りのない正しいフローチャートが容易に作成できるという効果がある。

【0044】〈具体例2〉上記のようにフローチャートを作成していく場合に、機能パーツを相互に接続しあるいは機能パーツの各部に別の機能パーツを挿入する作業が進められる。ここでは、ユーザによる機能パーツの挿入位置が正しいかどうかを判断し、フローチャートの作成を支援する例を説明する。

【0045】図6には、これを実現するための装置ブロック図を示す。この図は、図1に示した装置の構成を基準とし、同一部分には同一符号を付している。この具体例2の装置には、図1に示した装置に挿入判定部31を追加している。

【0046】図7には、挿入判定用表示例説明図(その1)を示す。図の(a)は、入口端子の機能パーツと出口端子の機能パーツを組み合わせたものである。実際には、この間に様々な機能パーツが挿入されてフローチャートが完成する。図に示す入口端子と出口端子の中間に○印を表示している。この○印は、予め他の機能パーツを挿入して配置することができる位置を表示するためのもので、本発明では空パーツと呼ぶことにする。

【0047】各機能パーツには、予めこのような空パーツを含めておき、パーツパレットからドラッグしてきた機能パーツをドロップすることができるポイントをガイドする。なお、この○印で示した空パーツと重ねて表示されている×印は、後で説明するフローチャート中の矢印の先を接続することができる接続点マークを表す。

【0048】図の(b)には、(a)のフローチャート中にいくつかの機能パーツが挿入された新たなフローチャートを示す。図中の、例えば破線の○印は、イベント2を含むステップ中に別のイベントを挿入することができる空パーツである。また、◎の空パーツは、各機能パーツの入り口や出口に配置されている。○印の空パーツは、各機能パーツの間に配置されている。これらの、○印や◎印、破線の○印といった空パーツは、必要に応じてそれぞれ挿入することができる機能パーツの種類を区別するために色分けされたり、形状を変えて表示されたものである。

【0049】そして、ユーザがパーツをドラッグしている間、マウスカーソルは、ドロップ可能な位置にいるときと、不可能な位置にいるときで、その形状を変化させる。こうして、ユーザに機能パーツのドロップ位置を意識させることができる。また、ドロップ不可能な位置にユーザが機能パーツをドロップしようとしても、システムはその操作を無視する。こうした動作自体は、既存のソフトウェアにより既に広く実施されており、容易に実現できる。

【0050】次に、本発明の装置の動作を説明する。まず、ユーザは入力装置1を使って、図2に示したようなパーツパレット12中の機能パーツをドラッグする。入力部2は、機能パーツのドラッグ開始を検出する。入力部2は、ドラッグ中のパーツの位置を順次検出して、その位置情報を挿入判定部31に渡す。挿入判定部31は、ドラッグ中のパーツ位置が空パーツの上にあるかどうかを判定する。そして、その結果をフローチャート表示部4に通知する。フローチャート表示部4は、入力部2の指示に従ってドラッグ中のマウスカーソルの形状を変更して、ユーザにその位置がドロップ可能な位置かどうかを通知する。

【0051】機能パーツがドロップされた場合、入力部2はそれを検知し、ドロップされた位置を挿入判定部31に通知する。挿入判定部31は、ドロップされた位置が空パーツ上かどうかを判断し、不適切であればその操

作を無視する。また、挿入判定部31は、ドロップされた位置が空パーツ上であれば、フローチャート上の該当位置にパーツが挿入されたことをフローチャート編集部3に通知する。

【0052】フローチャート編集部3は、機能パーツライブラリ6から機能パーツのデータを取り出し、フローチャートデータ7の該当する部分にパーツデータを挿入する。フローチャート編集部3は、新たな機能パーツが挿入されたことによって、新たに必要となる空パーツをフローチャートデータ7に挿入する。フローチャート表示部4は、フローチャートデータ7が更新されたことを検出すると、その結果を表示装置5に出力しフローチャートを更新して表示する。

【0053】従来、例えばドロ系ツールの用いてフローチャートを作成した場合には、図形パーツを自由に配置することができ、それらのパーツに正しい前後関係を与えるのはユーザの責任に任されていた。従って、ユーザのレベルによって不適切なパーツ配置を行うことも少なくなかった。

【0054】この具体例では、常に適切な位置にのみパーツが挿入されるように挿入可能な箇所を指定し、これを挿入判定部31において判定する。従って、例えば挿入した機能パーツがどこにもつながらずに浮いてしまうといった誤りが防止でき、常にプログラムの正しいフローチャートの入力ができる。更に、機能パーツを挿入したポイントにより、挿入された機能パーツがプログラム上のどの部分に挿入されたかをも認識することが可能になる。この動作を具体的に説明する。

【0055】図8には、挿入判定用表示例説明図(その2)を示す。図の(a)に示すように、このフローチャートがステップ1という機能パーツとステップ2という機能パーツとを接続して構成されているものとする。このとき、一般には、図中の(1)という部分に挿入した処理は、ステップ1のイベント処理なのか、ステップ2を実行する前の初期処理なのかを判別することができない。

【0056】ところが、既に説明したように、例えば各機能パーツの入口と出口に◎の空パーツを配置しておく。そして、(b)に示すように、ステップ1とステップ2の2組の機能パーツを接続した場合、その接続点に新たな○印の空パーツを加える。(2)の部分にある◎印の空パーツはステップ1に属する処理である。(3)の部分にある◎印の空パーツはステップ2に属する処理である。そして、○印の空パーツは新たな機能パーツを挿入することができる部分である。これを利用することによって、この部分に新たに挿入された機能パーツがステップ1あるいはステップ2に属する処理なのか、独立した処理なのかを区別することができる。

【0057】従って、各機能パーツを、既に説明した要領でソースコードに置き換え、ソースプログラムを生成

10

20

30

40

50

する場合に、新たに挿入された機能パーツのソースコードをどの機能パーツのソースコードに含めるかを判断することが可能になる。もちろん、周辺の空パーツを表示装置 5 に表示し続けられれば、フローチャート作成者にも、その区別ができる。

【0058】〈具体例 2 の効果〉以上のように、フローチャート中に機能パーツを挿入することができる場所を限定し、挿入判定部 3 1 によってこれを判定するように制御すれば、誤った場所に機能パーツを挿入するといったミスを防止できる。また、プログラムの流れを明確にし、機能パーツ単位のソースプログラムへの自動変換が容易になる。

【0059】〈具体例 3〉具体例 2 を用いて説明したように、各機能パーツに予め他の機能パーツを挿入できる位置を空パーツとして設定しておけば、フローチャートの作成が容易になるが、挿入可能な機能パーツを制限すれば、より誤りの無いフローチャートの入力が可能になる。そこで、挿入可能な機能パーツを判定するためのデータを設定する。

【0060】図 9 には、具体例 3 の装置ブロック図を示す。この装置は、具体例 2 の装置に対し、挿入判定マトリクスを新たに追加したものである。その他の部分は具体例 2 の装置と同一の構成であり、同一の符号を用いて表現した。

【0061】図 10 (a) には、挿入判定マトリクスの内容説明図を示す。また、(b) には、挿入判定マトリクスを参照して動作する挿入判定部の動作説明図を示す。図 10 の (a) に示すように、左側に示した空パーツは、それぞれ○、◎、破線の○及び太線の○というように区別されている。これらの空パーツがフローチャート中に存在する場合には、それぞれ図の最上段に示した様々な機能パーツ例えばステップ、分岐、イベント処理、端子、前判定、後判定、ケース、結合子等の機能パーツのうち、○が付いたものだけをそこに挿入することができる。

【0062】図の (a) に示す例では、例えば破線の○印はイベント以外の挿入を禁止している。この破線の○印は、図 7 (b) に示すようなステップの機能パーツ中表示されていた。ステップは、イベント待ちの状態を示すもので、一般的なループ処理とは異なる。従って、ステップ中にはイベント以外を挿入することができない。ユーザは、このような空パーツの形状や種類によって、ドラッグしている機能パーツの挿入可能な位置を見分けることができる。具体的には次のようにして装置が動作する。

【0063】まず、機能パーツがドロップされた場合、図 9 に示す入力部 2 はそれを検知し、ドロップされたパーツの種類とドロップされた位置情報を挿入判定部 3 1 に通知する。挿入判定部 3 1 は、ドロップされたパーツと空パーツを示す○印の種類が挿入可能な組み合わせか

どうかを判定する。これは、図 10 (a) に示したような挿入判定マトリクス 3 2 を参照して行う。

【0064】挿入判定マトリクス 3 2 で、妥当な組み合わせであるという表示がされていない場合には、ドロップ操作を無視する。挿入判定部 3 1 が判定した結果、妥当な組み合わせであれば、フローチャート上の該当位置に機能パーツが挿入されたことをフローチャート編集部 3 に通知する。フローチャート編集部 3 は、機能パーツライブラリ 6 から該当する機能パーツのデータを取り出し、フローチャートデータ 7 の該当位置に挿入する。そして、フローチャート編集部 3 は、パーツが挿入されたことによって新たに必要になる空パーツをフローチャートデータ 7 に挿入する。フローチャート表示部 4 は、その後フローチャートデータ 7 の更新を検出し、表示装置 5 に新たなフローチャートを表示する。

【0065】例えば、図 10 (b) において、破線で囲まれたステップの機能パーツが既に描かれており、イベント 1 とイベント 2 のパーツが挿入されていたとする。これに新たに処理のパーツを挿入すると、これはイベント待ちのステップとして意味をなさなくなる。ループの中に処理が挿入されることによって、タスクのウェイト状態への移行は不可能になってしまい、通常の繰り返し処理と同じになるからである。

【0066】更に、本来、ステップがあるべき場所にそのようなパーツが存在することによって、状態遷移図への変換も不可能になる。また、そのような不適切なパーツの挿入は、イベントドリブン指向としての正当性を損なうことになる。従って、図 10 (b) に示すようなフローチャートが作成できないようにすることによって、この発明の目的は達成される。

【0067】〈具体例 3 の効果〉以上に示すように、挿入判定マトリクスを設けることによって、空パーツの部分に挿入可能な機能パーツの種類を限定することができる。こうして不適切なパーツの挿入によって誤ったフローチャートの作成を防止できる。

【0068】〈具体例 4〉具体例 3 では、機能パーツを挿入すべき位置を判断した。プログラムのフローチャート入力作業では、機能パーツを挿入することの他に、イベント処理をした後の状態遷移先を矢印の接続がある。この矢印の接続に誤りがあれば論理的な矛盾が生じ、更にイベントドリブン指向としてのプログラムの正当性を損なう。

【0069】図 11 には、具体例 4 の装置のブロック図を示した。この装置には、矢印接続の正当性を判断する接続判定部 3 3 を設けた。この他の機能ブロックは、具体例 3 を用いて説明したものと同様であり、それぞれ同一の記号を付している。

【0070】図 12 は、本発明によって描かれたフローチャートの一例である。この図を用いて、接続判定動作の説明を行う。この図の右側に示した Then 処理と、

Else処理に続く部分には、(4)あるいは(5)に示すような矢印が設けられている。矢印の首部にある黒塗りの四角形をカーソルでクリックしこれをドラッグすることによって、矢印を任意の場所に移動できる。そして、指定した場所にドロップすると、その矢印の先端がフローチャート中に接続される。ここで、矢印の先端が接続可能な場所に×印の接続点マークを示した。

【0071】即ち、図の(1)、(2)、(3)以外の場所にはこれらの矢印の先端を接続することができない。即ち、プログラムのイベントドリブ的な性格を保持するために、矢印は、状態遷移先として適切な場所에만接続することができる。なお、図の(6)に示した矢印はステップの機能パーツに含まれる。この矢印の先を移動させると機能パーツ自身の正当な動作を妨げる。従って黒塗りの四角形が用意されておらず、その移動は認められない。図3(e)や(f)に示した前判定ループ、(f)に示した後判定ループの矢印も同様に接続変更ができない。

【0072】例えば、状態遷移以外の目的で矢印の先端を接続すると、イベントドリブ的な処理が不可能になる。このために以上のような制限が加えられる。そして、フローチャート入力中に許可されない接続を行おうとした場合には、矢印の移動は認められず元の場所に戻る。

【0073】図12に示す例では、(1)を起点とするステップの機能パーツは、2種のイベントに従って状態遷移する。上側に配置されたイベント1により状態遷移をする場合、図の右側に示す条件分岐の機能パーツによる処理に進む。条件を満たせばThen処理、条件を満たさない場合にはElse処理が実行される。

【0074】一方、(1)で始まるステップの下側のイベント2で状態遷移する場合、(2)の直上にあるイベント処理が実行されて、(2)の部分から始まるステップに進む。こうして、次のイベント3を待つ状態に遷移する。このため、(4)や(5)で示す矢印は、状態遷移の境界である(1)、(2)あるいは(3)以外の場所には移動できない。こうした基準によって、上記接続判定部33は動作する。

【0075】図13には、図12の(5)に示した矢印を(3)の位置まで移動した後の接続判定動作の説明図(その2)を示す。この図を用いて、図11に示した装置の動作を説明する。まず、ユーザは入力装置1を使って、フローチャート表示エリア13上に表示された

(5)の矢印先端をクリックする。そして、そのままドラッグする。入力部2は、入力装置1で矢印のドラッグが開始されたことを検知すると、接続判定部33にその旨を接続する。接続判定部33は、ドラッグされた矢印がドラッグ可能かどうかを判定し、(6)のようにドラッグが禁止された矢印であればその操作を無視する。

【0076】また、入力部2は、入力装置1からドラッ

グ中の矢印先端位置を順次検出し、その位置情報を接続判定部33に通知する。接続判定部33は、受け取った矢印の位置情報が接続可能な×印上にあるかどうかを判定する。そして、その結果をフローチャート表示部4に通知する。フローチャート表示部4は、入力部2の指示に従ってドラッグ中のマウスカソルの形状を変更し、ユーザに対しその位置への接続可否を通知する。

【0077】矢印先端がドロップされた場合、入力部2はそれを検出し、ドロップされた位置を接続判定部33に通知する。接続判定部33は、ドロップされた位置が適切な×印上かどうかを判断し、不適切であればその矢印をデフォルトの位置即ちドラッグ開始前の位置に再接続するよう、フローチャート編集部3に通知する。接続判定部33は、ドロップされた位置が適切な×印上であると判断したとき、その矢印を指定された位置に接続するようにフローチャート編集部3に指示する。

【0078】フローチャート編集部3は、フローチャートデータ7の矢印接続先を指示された位置になるように変更する。その後、フローチャート表示部4はフローチャートデータ7が更新されたことを検出して、表示装置5にフローチャートの再表示を指示する。

【0079】図14には、誤ったフローチャートの例を示す。この図に示すフローチャートは、条件分岐を行った結果の分岐先を示す矢印が前判定ループの中に飛び込んでいる。こうしたフローチャートは、goto文を使わざるを得ない。goto文の使用は、プログラムの可読性、保守性を損ねるとして、一般的にその使用を避けることが望ましいといわれている。

【0080】即ち、こうした流れは、状態遷移を中心としたイベントドリブン式構造化プログラミングの思想に反するため不適切である。従来のツールを用いた場合には、こうしたフローチャートを描くことは自由であり、これを禁止するのはユーザの意志に任されていた。この発明では、予め矢印を接続する場所が制限されて、接続判定部33がそれを判断するから、こうした不適切な矢印の接続を禁止できる。

【0081】〈具体例4の効果〉以上のように、この発明では、フローチャート中への不適切な矢印の接続を禁止でき、例えばユーザのプログラミング能力のレベルによらず、常に構造化されたgotoレスのフローチャートを作成するようにユーザ支援ができる。また、論理的に誤りのある接続を防止することもでき、その後のバグのないプログラムの自動生成を可能にする。

【0082】従って、たとえ上記のような構造化プログラミングによる制限に反しなくても、その矢印の接続を行った場合に、不適切な動作が発生するような全ての場合に、矢印の接続を禁止する処理に広く利用できる。

【0083】〈具体例5〉上記のような本発明の装置において、フローチャートデータは、機能パーツの組み合わせにより表現できる。従って、例えば機能パーツの名

称と相互の接続関係を明らかにすれば、フローチャートデータを生成することができる。また、こうしたフローチャートがあれば、各機能パーツを対応するソースコードに置き換え、機能パーツと状態遷移をする先を示す矢印をソースコード化することによって、自動的なソースプログラム生成が可能となる。

【0084】しかしながら、表示装置に機能パーツを表示する場合には、機能パーツを構成する各図形部品のイメージを適切にリンクさせ、変形に対応する必要がある。こうした場合に、各図形パーツの効率的な確な管理が必要となる。そこで、この具体例では、以下に説明するような階層構造のフローチャートデータを用いる。

【0085】図15に、具体例5の装置ブロック図を示す。ここに示すブロック図は、具体例4の装置に対し、階層構造フローチャートデータ34を採用した点異なる。その他の部分は具体例4と同一符号を付した。

【0086】次の図16、図17、図18を用いて、階層構造フローチャートデータの構成を説明する。図16は、ステップ1とステップ2とステップ3の状態を持つプログラムのフローチャートを示す。このフローチャートには、ステップ1から直接ステップ3へ状態遷移する動作とステップ1からステップ2を経てステップ3へ状態遷移をするプログラムが示されている。ここで、ステップ1からステップ2へ状態遷移する過程で実行されるイベント処理の直前の(1)の部分に、新たな機能パーツを挿入することを考える。

【0087】図17には、図16中の(1)の部分に分岐条件処理を行う機能パーツを追加したところを示す。図16に示すフローチャートでは、ステップ1の状態からイベント1-1が発生した場合に、対応するイベント処理を行ってステップ2に進む。また、イベント1-2が発生した場合に、対応するイベント処理を実行して、ステップ3に進む。ステップ2の状態では、イベント2-1が発生した場合に対応するイベント処理を実行し、ステップ3に進む。ステップ3は、イベント3-1が発生した場合に、このプログラムを終了する処理を行う。図17では、イベント1-1が発生した場合に、対応するイベント処理を実行する前に、新たな機能パーツにより分岐条件判断を行う。フローチャート入力作業中に、こうした機能パーツの追加やその後の削除を行うとき、その表示制御に適する構造のフローチャートデータが存在することが望ましい。

【0088】図18には、図16のフローチャートを階層構造のフローチャートデータで示した例を図示した。まず、このフローチャートは、スタートの端子とストップの端子との間に子パーツが挟まれた基本構造をしている。これがNo. 1のデータ階層に表示されている。No. 1のデータ階層に示した子パーツは、図16に示すように、ステップ1に示した機能ステップと、ステップ2に示した機能ステップと、ステップ3に示した機能ス

テップにより構成される。これらのステップを破線の枠内に図示した。

【0089】そして、それぞれのステップ毎に、状態遷移先を矢印で示している。図中、赤丸と表示したのは、各ステップの入口や出口に配置した空パーツである。イベントドリブン形式のプログラムでは、基本的にこれだけで全ての処理が表現できる。上の階層と下の階層との関係は、後者が前者の属性データを示すという関係である。矢印は、ポインタデータである。例えば、スタートの端子とストップの端子の属性データは、No. 2の階層部分に表示される。即ち、いずれも楕円とフローチャートの一部となる線とによって構成されるという内容になる。

【0090】次に、No. 3の階層について説明を行う。ここでは、ステップ1の属性データについてのみその例を示し、その他については同様であるから図示を省略した。ステップ1の機能ステップには、図16に示すように、スタートの端子に接続される直線と、先頭の破線の円で示した空パーツと、イベント1-1、空パーツ、イベント1-2、空パーツといった図形パーツが存在する。さらに、イベント待ち状態を表現するループ線が存在する。

【0091】そこで、図18に示すNo. 3の階層には、先頭の線と、入口にある◎の空パーツに対応する子パーツと、イベント1-1へ進む線と、イベント1やイベント2を含めた子パーツと、ループ線という属性データが表示されている。◎の空パーツを示す子パーツは、No. 4の階層において、緑の丸印というようにその属性が表示されている。また、No. 4の階層に3つの青の○印と、これらに挟まれたイベント1-1やイベント1-2が、No. 3の子パーツを示す属性データとなっている。

【0092】また、イベント1-1やイベント1-2は、それぞれ菱形の図形と、分岐線と、更にその分岐線の先に連なる子パーツとから構成されている。これをNo. 5の階層に示した。そして、図16に示す(1)のイベント処理は、緑の○印で示される空パーツの間に処理の機能パーツが挟まれた構成となっている。これをNo. 6の階層に示した。また、処理の機能パーツは四角の図形から構成されるから、これをNo. 7の階層に示した。

【0093】以上のように、このフローチャートデータは、状態を示すステップの情報と、これらのステップ間の状態遷移を示す矢印の情報を基本となるデータ階層に表現し、各状態を示すステップ毎に、そのステップの属性データを順次階層構造で表現している。このような構成にした場合に、図16に示す(1)の部分に図17に示すような(2)の機能パーツを挿入したとき、図19に示すように、フローチャートデータが書き換えられ

【0094】即ち、図19には、図18に示したNo. 6の階層中に、新たにNo. 5の階層で示した子パーツに属する「分岐」と「緑の○で表す空パーツ」を追加している。そして、「分岐」の属性を表すために、No. 7の階層には、その形状が菱形であって、線と2つの子パーツと分岐線とにより構成されることが示されている。また、2つの子パーツは、別の新たな機能パーツを挿入できる緑の○印で示した空パーツであると表現している。

【0095】このように、新たな機能パーツを追加する場合には、階層構造になった機能パーツのデータを、階層構造になったフローチャートの該当箇所へ一括して挿入すればよい。逆に、こうした機能パーツを外す場合には、図19中の破線で示した階層構造の機能パーツ要素を一括して削除すればよい。即ち、「分岐」という機能パーツに必要な全ての部品が階層構造となり、フローチャートも階層構造になっているので、機能パーツを追加したり削除をしたりする場合に、一部の図形パーツが取り残されてしまったりする矛盾が発生しない。

【0096】〈具体例5の効果〉以上説明したように、フローチャートデータを階層構造で表現すれば、機能パーツ単位でフローチャートを作成処理する作業の際に、機能パーツを追加したり削除したりして試行錯誤する作業の際、フローチャートデータの取り扱いを容易にし、誤りの発生を防止できる。従って、フローチャートに矛盾を発生させたり、意味のないパーツを残したりするという問題がない。しかも、パーツを削除した際、自動的にフローチャート中の該当部分の前後を接続することが容易になる。

【0097】〈具体例6〉図18に示したフローチャートデータのNo. 2の階層に着目する。この階層には、図に示したように、状態を示すステップの情報とこれらのステップ間の状態遷移を表す矢印の情報のみが現れている。こうしたデータを利用すれば、プログラムの基本的な構造解析を行うことができる。

【0098】図20は、この構造解析のための機能ブロックを図示した。図1に示した装置にこうした機能ブロックを追加することによって、フローチャートデータから、例えば図21に示すような状態遷移図を生成し表示することができる。即ち、図20に示す装置は、階層構造フローチャートデータ34と、変換部36と、状態遷移図データ37、データ表示部38及び表示装置5から構成される。さらに、この図には、階層構造フローチャートデータ34や機能パーツライブラリ6を参照して、フローチャートに対応するソースプログラムを生成するプログラム生成部を図示した。

【0099】プログラム生成部40は、まず、上記階層構造フローチャートデータあるいはその他の構造のフローチャートデータを参照して、各状態毎に、イベントの内容とその属性データを生成する。次に、機能パーツラ

イブラリ6を参照して、フローチャート中に使用されている機能パーツの既に説明したようなソースコードを取得する。そして、上記イベントや状態の属性データに付随したプログラムコード記述部分にそのソースコードをコピーする。これで、プログラムの自動生成ができる。これ以外の具体的なプログラム生成手順は、従来のCASEツールで実行されている処理と同様である。

【0100】階層構造フローチャートデータ34は、図18に示すような内容のデータである。変換部36は、例えばそのNo. 2のデータ階層を参照して、そのデータを変換処理し、状態遷移図データ37を生成する部分である。データ表示部38は、この状態遷移図データ37を表示装置5に表示する部分である。ソフト技術者がプログラム全体の動作を解析する場合、一般にその状態遷移図を作成する。これを自動的に作成し、表示装置に表示することができれば、フローチャート作成後の結果をユーザが点検し、あるいは他のソフトウェア技術者が内容を理解するのに役立つ。

【0101】図21は、状態遷移図の例説明図である。この図を参照しながら、図20に示した装置の動作を説明する。まず、始めに変換部36は、フローチャートデータ34を解析し、図18に示したNo. 2のデータ階層から状態を示すステップを抽出する。また、変換部36は、図18に示すフローチャートデータを解析し、各ステップの状態を表示装置に表示する際の位置やシンボル図形等を決定する。図21中の円内に「ステップ1」、「ステップ2」、「ステップ3」と書いたものがそのシンボル図形である。

【0102】次に、変換部36は、図18に示すNo. 2のデータ階層から矢印に相当するデータを抽出し、各状態を表示する、これには、例えば図21に示す円の間を矢印のイメージでつなぐ。更に、図18に示すNo. 3以下のデータ階層のイベントを抽出し、状態遷移を表す矢印にそれぞれその意味を示す情報を付加する。同時に、図18に示したNo. 1の階層を参照し、スタートやストップの端子に相当する情報を付け加える。このようにして、図21に示すようなイメージを生成し、データ表示部38の制御によって表示装置5にその図形イメージを表示する。

【0103】〈具体例6の効果〉以上説明したように、フローチャートデータを上記のような構成にすることによって、状態遷移を表すデータ階層から必要な状態を抽出し、該当するイベントから状態遷移の内容を抽出して、自動的に状態遷移図を生成することができる。これによって、フローチャートがメカトロニクス製品に実装されているプログラムに属したイベントドリブン型のプログラムであるかどうかの検証を即座に行うことができ、フローチャートの作成やその解析処理に有効に活用される。

【0104】なお、上記の具体例1から具体例6に示し

た各発明は、初めに説明したように、それぞれパーソナルコンピュータやワークステーション等にインストールしたプログラムによって実行することができる。そして、これらのプログラムは、フロッピーディスクやCD-ROM、あるいはその他のプログラム記録媒体に格納することができる。更に、通信回線等を利用してコンピュータに直接インストールして実施することが可能である。

【図面の簡単な説明】

【図1】本発明のフローチャート入力装置を示すブロック図である。

【図2】表示装置に表示されたウインドウの概略図である。

【図3】機能パーツの具体例説明図である。

【図4】ソフトウェア設計例説明図である。

【図5】フローチャート作成とプログラムへの変換例説明図である。

【図6】具体例2の装置ブロック図である。

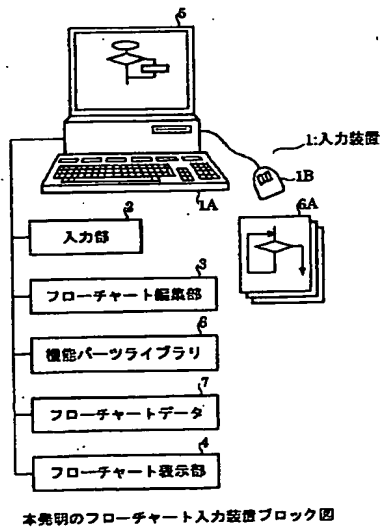
【図7】挿入判定用表示例説明図（その1）である。

【図8】挿入判定用表示例説明図（その2）である。

【図9】具体例3の装置ブロック図である。

【図10】挿入判定マトリクスの説明図である。

【図1】



【図11】具体例4の装置ブロック図である。

【図12】接続判定動作の説明図（その1）である。

【図13】接続判定動作の説明図（その2）である。

【図14】誤ったフローチャート例の説明図である。

【図15】具体例5の装置ブロック図である。

【図16】変更前のフローチャート例説明図である。

【図17】変更後のフローチャート例説明図である。

【図18】フローチャートデータ例（変更前）の説明図である。

【図19】フローチャートデータ例（変更後）の説明図である。

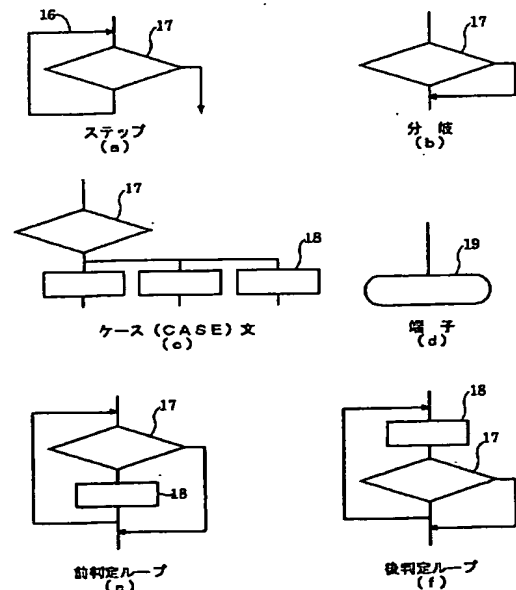
【図20】具体例6の装置のブロック図である。

【図21】状態遷移図の例説明図である。

【符号の説明】

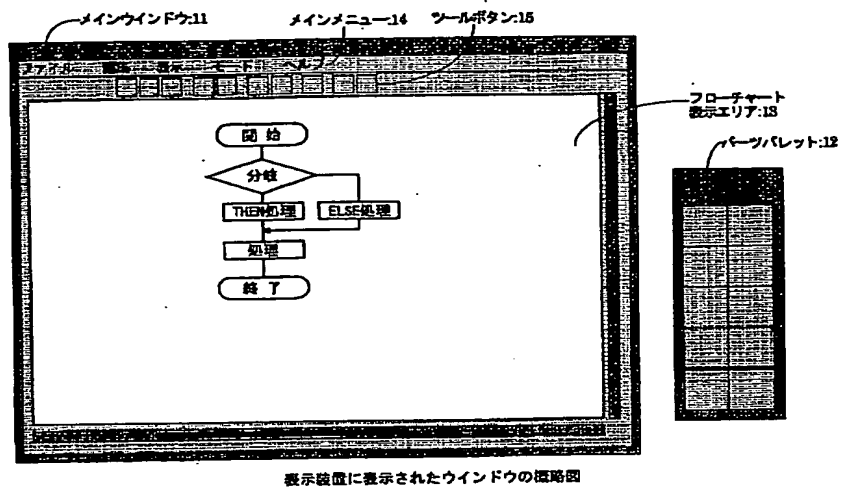
- 1 入力装置
- 2 入力部
- 3 フローチャート編集部
- 4 フローチャート表示部
- 5 表示装置
- 6 機能パーツライブラリ
- 6A 機能パーツ
- 7 フローチャートデータ

【図3】

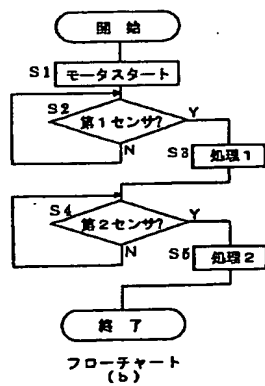
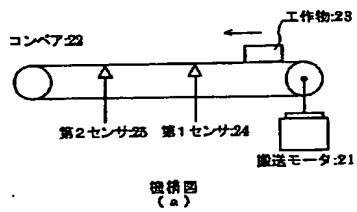


機能パーツの具体例説明図

【図2】

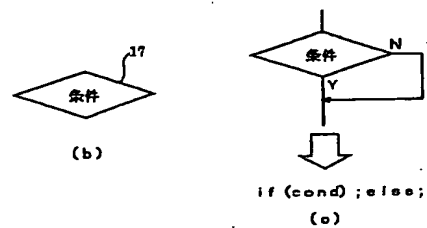
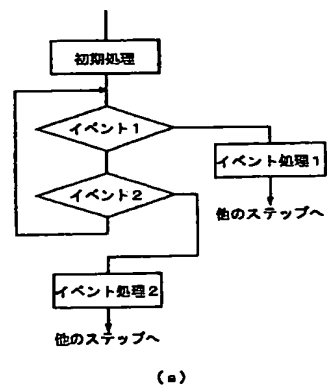


【図4】



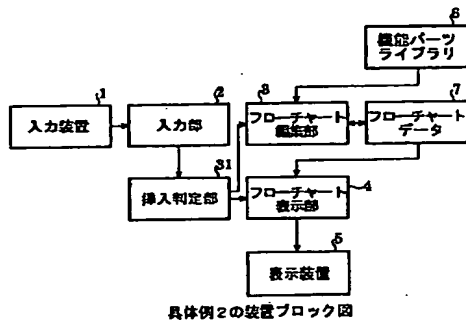
ソフトウェア設計例説明図

【図5】

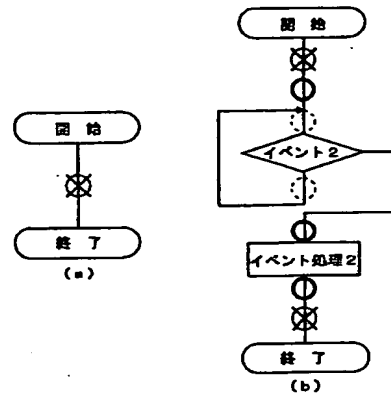


フローチャート作成とプログラムへの変換例

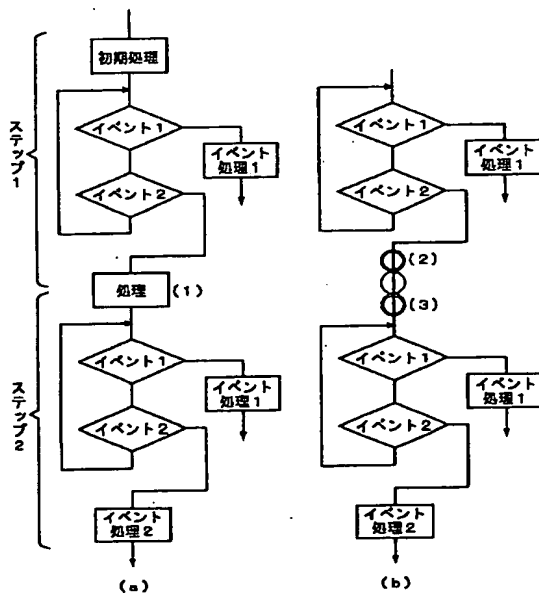
【図6】



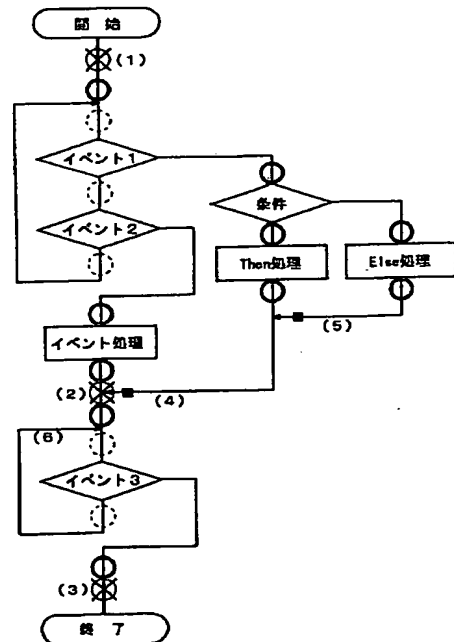
【図7】



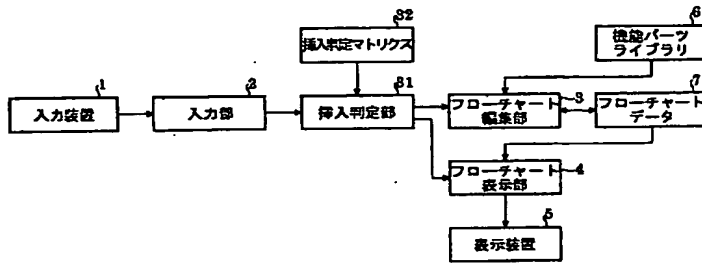
【図8】



【図12】

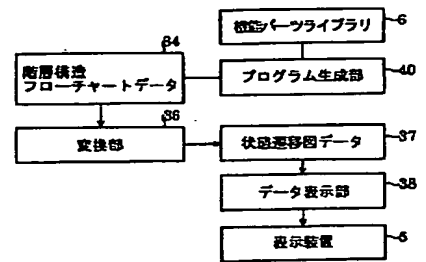


【図9】



具体例3の装置ブロック図

【図20】

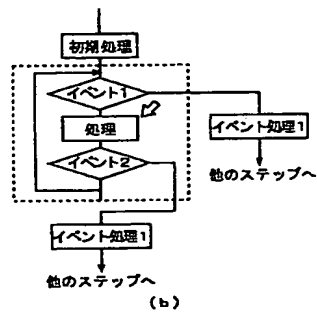


具体例6の装置のブロック図

【図10】

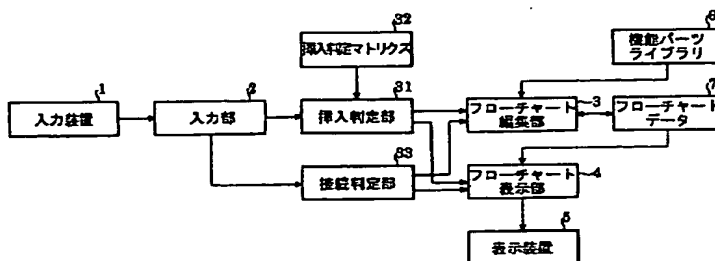
	Step	分岐	イベント	処理	端子	前判定	後判定	ケース	結合子
○	×	○	×	○	×	○	○	○	×
●	○	×	×	×	○	×	×	×	○
破線○	×	×	○	×	×	×	×	×	×
○	×	○	×	×	×	×	×	×	×

(a)



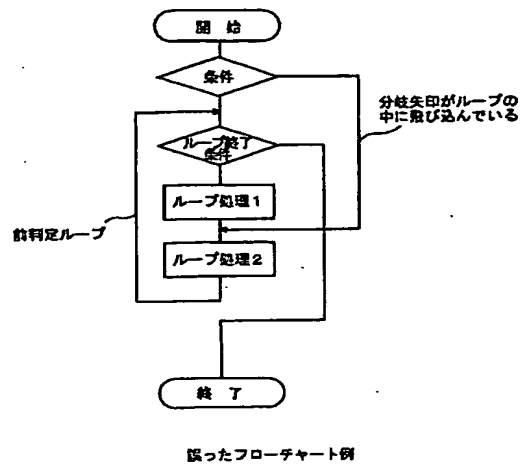
挿入判定マトリクスの説明図

【図11】



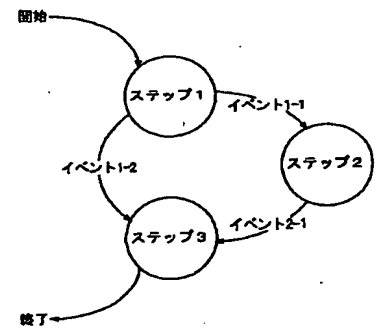
具体例4の装置ブロック図

【图 14】



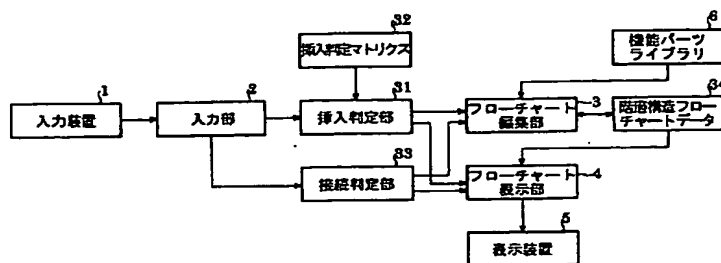
接続判定動作の説明図（その２）

【图 2 1】



状態遷移図の例説明図

【図 15】



具体例5の装置ブロック図

【图 17】



【图 18】

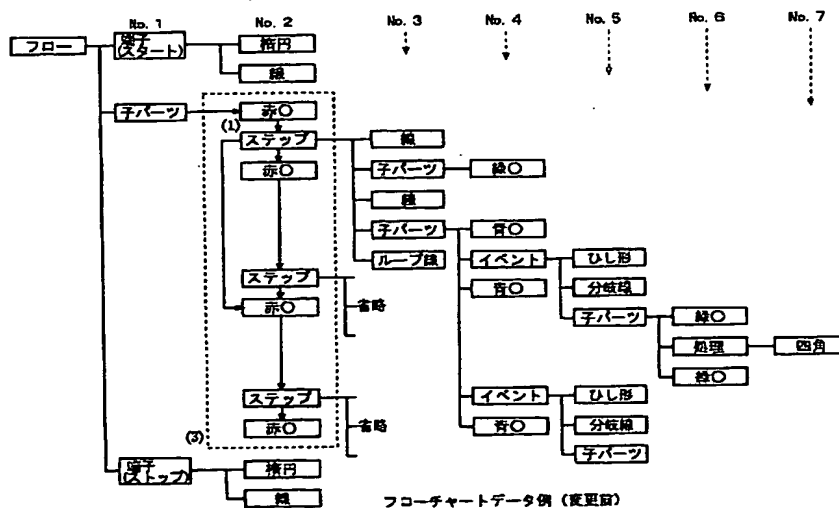


Figure 1 is a flowchart diagram illustrating a sequence of steps. The steps are labeled as follows:

- No. 1: ステップ (Step)
- No. 2: 線 (Line)
- No. 3: 子パーツ (Sub-part)
- No. 4: 線〇 (Line O)
- No. 5: 青〇 (Blue O)
- No. 6: イベント (Event)
- No. 7: 青〇 (Blue O)
- No. 8: ひし形 (Rhombus)

The flowchart shows a main vertical path with branches. From No. 1, a branch leads to No. 2. From No. 2, a branch leads to No. 3. From No. 3, a branch leads to No. 4. From No. 4, a branch leads to No. 5. From No. 5, a branch leads to No. 6. From No. 6, a branch leads to No. 7. From No. 7, a branch leads to No. 8. The flowchart also includes dashed lines indicating connections between steps.